Building httpd

While many pre-built binaries are available for Apache httpd, it can sometimes be advantageous to build your own version - for example, if you want a different selection of modules, or you want to run a specific version. This page documents some of the issues you may come across when building your own version.

General

The Apache httpd build process is based on gnu autotools. This means that you will have a **configure** script. At its simplest, configuring, building, and installing can be done with the following commands:

```
> ./configure

> make

> make install
```

In most cases however, you will need additional configuration options to build exactly the features you need, and remove the ones you do not. Some of the available features and configure options are described below.

Build Options

Proxy

If you are planning on using the server as an HTTP proxy server, you will need to use the **`--enable-proxy`** and **`--enable-proxy-http`** options (other options are available for other proxy types). A simple proxy setup can be implemented using the following directives:

```
ProxyPass        /front/end/path/    http://internal-host/back/end/path/

ProxyPassReverse /front/end/path/    http://internal-host/back/end/path/
```

This will forward all requests that come in to the Apache server with the path '`/front/end/path/`' to another web server on the host '`internal-host`' with the path '`/back/end/path/`'. For more details on proxy configuration, refer to the proxy module documentation.

SSL

If you wish to provide secure access to your browser, make sure to enable **`--enable-ssl`** (this module is required if you wish to run the Apache URS Authentication Module). This will enable building and installation of the SSL module. Note that enabling SSL over HTTP (i.e. HTTPS) requires more CPU cycles (for both client and server) than a plain HTTP connection, and introduces some more latency, since it involves data encryption and protocol negotiation. Many modern clients can submit multiple requests over a single connection (permitted by version 1.1 of the HTTP protocol) which can reduce the latency issue for web servers that serve html pages consisting of multiple items (images, text, css, etc). For plain data servers (e.g. large hdf files), the additional latency is likely to be negligible compared to the data transfer time.

Check the SSL module documentation for further details.

Compression

If your server is bandwidth limited and has spare CPU cycles, enabling data compression on output can help. Using the **`--enable-deflate`** option will enable the mod_deflate module. This uses an output filter to compress the output prior to sending it over the network. This will only occur for clients that support it (as indicated using the **`Accept-Encoding`** request header). Note that it is not a good idea to try and compress data that is already compressed (e.g. jpeg images, .gz files, etc). In this case, you may wish to use the AddOutputFilterByType directive, e.g.

```
AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css text/javascript
application/javascript
```

Check the deflate module documentation for further details.

Caching

Caching is not enabled by default, but it can be enabled using **`--enable-cache`** and one of **`--enable-mem-cache`** or **`--enabled-disk-cache`**. Caching is generally useful for content that may take time to assemble - e.g. CGI generated content or proxy content. It can speed up access at the cost of memory (or local disk). Memory caching provides the best performance enhancement, but can require significant amounts of memory since it works on a per-process basis - each worker process (see the MaxClients configuration parameter) requires its own cache. This can be mitigated to some extent by using the worker MPM model. The performance of disk caching will vary depending upon many factors, including the disk performance and available IO bandwidth. Ideally, any disk cache should be placed on a separate disk to your main data for optimal performance. Note that the OS may provide in memory file buffering - effectively giving you the benefits of a single memory cache for all worker processes, however you would need to run tests in order to accurately asses the performance gains for your situation. Check the cache module documentation for further details.

Warning - page still under construction!

ToDo

- cgi
- rewrite